

A Note on Monitors and Büchi automata

Volker Diekert¹ and Anca Muscholl² and Igor Walukiewicz²

¹ Universität Stuttgart, FMI, Germany

² LaBRI, University of Bordeaux, France

Abstract. When a property needs to be checked against an unknown or very complex system, classical exploration techniques like model-checking are not applicable anymore. Sometimes a monitor can be used, that checks a given property on the underlying system at runtime. A monitor for a property L is a deterministic finite automaton \mathcal{M}_L that after each finite execution tells whether (1) every possible extension of the execution is in L , or (2) every possible extension is in the complement of L , or neither (1) nor (2) holds. Moreover, L being monitorable means that it is always possible that in some future the monitor reaches (1) or (2). Classical examples for monitorable properties are safety and cosafety properties. On the other hand, deterministic liveness properties like “infinitely many a ’s” are not monitorable.

We discuss various monitor constructions with a focus on deterministic ω -regular languages. We locate a proper subclass of of deterministic ω -regular languages but also strictly large than the subclass of languages which are deterministic and codeterministic; and for this subclass there exists a canonical monitor which also accepts the language itself.

We also address the problem to decide monitorability in comparison with deciding liveness. The state of the art is as follows. Given a Büchi automaton, it is PSPACE-complete to decide liveness or monitorability. Given an LTL formula, deciding liveness becomes EXPSPACE-complete, but the complexity to decide monitorability remains open.

Introduction

Automata theoretic verification has its mathematical foundation in classical papers written in the 1950’s and 1960’s by Büchi, Rabin and others. Over the past few decades it became a success story with large scale industrial applications. However, frequently properties need to be checked against an unknown or very complex system. In such a situation classical exploration techniques like model-checking might fail. The model-checking problem asks whether all runs satisfy a given specification. If the specification is written in monadic second-order logic, then all runs obeying the specification can be expressed effectively by some Büchi automaton (BA for short). If the abstract model of the system is given by some finite transition system, then the model-checking problem becomes an inclusion problem on ω -regular languages: all runs of the transition system must be accepted by the BA for the specification, too. In formal terms we wish to check $L(\mathcal{A}) \subseteq L(\varphi)$ where \mathcal{A} is the transition system of the system and φ is a formula

for the specification. Typically testing inclusion is expensive, hence it might be better to check the equivalent assertion $L(\mathcal{A}) \cap L(\neg\varphi) = \emptyset$. This is a key fact, because then the verification problem becomes a reachability problem in finite graphs.

Whereas the formulas are typically rather small, so we might be able to construct the Büchi automaton for $L(\neg\varphi)$, the transition systems tend to be very large. Thus, “state explosion” on the system side might force us to use weaker concepts. The idea is to construct a “monitor” for a given specification. A monitor observes the system during runtime. It is a finite deterministic automaton with at most two distinguished states \perp and \top . If it reaches the state \perp , the monitor stops and raises an “alarm” that no continuation of the so far observed run will satisfy the specification. If it reaches \top , the monitor stops because all continuations will satisfy the specification. Usually, this means we must switch to a finer monitor. Finally, we say that a language is monitorable, if in every state of the monitor it is possible to reach either \perp or \top or both.

The formal definition of monitorable properties has been given in [17] by Pnueli and Zaks. It generalizes the notion of a *safety property* because for a safety property some deterministic finite automaton can raise an alarm \perp by observing a finite “bad prefix”, once the property is violated. The extension to the more general notion of monitorability is that a monitorable property gives also a positive feedback \top , if all extensions of a finite prefix obey the specification. Monitors are sometimes easy to implement and have a wide range of applications. See for example [12] and the references therein. Extensions and applications for stochastic automata have been proposed in Sistla et al., see [6,19].

In the present paper we discuss various monitor constructions. A monitor for a safety property L can have much less states than the smallest DBA accepting L . For example, let $\Sigma = \{a, b\}$ and $n \in \mathbb{N}$. Consider the language $L = a^n b a \Sigma^\omega \setminus \Sigma^* b b \Sigma^\omega$. The reader is invited to check that L is a safety property and every DBA accepting L has more than n states. But there is monitor with three states, only. The monitor patiently waits to see an occurrence of a factor bb and then switches to \perp . Hence, there is no bound between a minimal size of an accepting DBA and the minimal size of a possible monitor. This option has been actually one of the main motivations to introduce the notion of monitor.

There are many deterministic languages which are far away from being monitorable. Consider again $\Sigma = \{a, b\}$ and let L be the deterministic language of “infinitely many a ’s”. It is shown in [4] that L cannot be written as any countable union of monitorable languages. On the other hand, if L is monitorable and also accepted by some DBA with n states and a single initial state, then there is monitor accepting L with at most n states.

In the last section of this paper we discuss the question how to decide whether a language is monitorable and its complexity. If the input is a Büchi automaton, then deciding safety, liveness, or monitorability is PSPACE-complete. If the input is an LTL formula, then deciding safety remains PSPACE-complete. It becomes surprisingly difficult for liveness: EXPSpace-complete. For monitorability the

complexity is wide open: we only know that is PSPACE-hard and that it can be solved in EXPSPACE.

1 Preliminaries

We assume that the reader is familiar with the basic facts about automata theory for infinite words as it is exposed in the survey [23]. In our paper Σ denotes a finite nonempty alphabet. We let Σ^* (resp. Σ^ω) be the set of finite (resp. infinite) words over Σ . Usually, lower case letters like a, b, c denote letters in Σ , u, \dots, z denote finite words, 1 is the empty word, and α, β, γ denote infinite words. By language we mean a subset $L \subseteq \Sigma^\omega$. The complement of L w.r.t. Σ^ω is denoted by L^c . Thus, $L^c = \Sigma^\omega \setminus L$.

A *Büchi automaton* (BA for short) is a tuple $\mathcal{A} = (Q, \Sigma, \delta, I, F)$ where Q is the nonempty finite set of states, $I \subseteq Q$ is the set of initial states, $F \subseteq Q$ is the set of final states, and $\delta \subseteq Q \times \Sigma \times Q$ is the transition relation. The accepted language $L(\mathcal{A})$ is the set of infinite words $\alpha \in \Sigma^\omega$ which label an infinite path in \mathcal{A} which begins at some state in I and visits some state in F infinitely often. Languages of type $L(\mathcal{A})$ are called ω -regular.

If for each $p \in Q$ and $a \in \Sigma$ there is at most one $q \in Q$ with $(p, a, q) \in \delta$, then \mathcal{A} is called *deterministic*. We write DBA for deterministic Büchi automaton. In a DBA we view δ as a partially defined function and we also write $p \cdot a = q$ instead of $(p, a, q) \in \delta$. Frequently it is asked that a DBA has a unique initial state. This is not essential, but in order to follow the standard notation $(Q, \Sigma, \delta, q_0, F)$ refers to a BA where I is the singleton $\{q_0\}$.

A *deterministic weak Büchi automaton* (DWA for short) is a DBA where all states in a strongly connected component are either final or not final. Note that a strongly connected component may have a single state because the underlying directed graph may have self-loops. A language is accepted by some DWA if and only if it is deterministic and simultaneously codeterministic. The result is in [21] which in turn is based on previous papers by Staiger and Wagner [22] and Wagner [25].

According to [17] a *monitor* is a finite deterministic transition system \mathcal{M} with at most two distinguished states \perp and \top such that for all states p either there exist a path from p to \perp , or to \top , or to both. It is a *monitor for an ω -language $L \subseteq \Sigma^\omega$* if the following additional properties are satisfied:

- If u denotes the label of a path from an initial state to \perp , then $u\Sigma^\omega \cap L = \emptyset$.
- If u denotes the label of a path from an initial state to \top , then $u\Sigma^\omega \subseteq L$.

A language $L \subseteq \Sigma^\omega$ is called *monitorable* if there exists a monitor for L . Thus, even non regular languages might be monitorable. If a property is monitorable, then the following holds:

$$\forall x \exists w : xw\Sigma^\omega \subseteq L \vee xw\Sigma^\omega \cap L = \emptyset. \quad (1)$$

The condition in (1) is not sufficient for non-regular languages: indeed consider $L = \{a^n b^n a \mid n \in \mathbb{N}\}\Sigma^\omega$. There is no finite state monitor for this language.

In the present paper, the focus is on monitorable ω -regular languages. For ω -regular languages (1) is also sufficient; and Remark 2 below shows an equivalent condition for monitorability (although stronger for non-regular languages).

The common theme in “automata on infinite words” is that finite state devices serve to classify ω -regular properties. The most prominent classes are:

- *Deterministic properties*: there exists a DBA.
- *Deterministic properties which are simultaneously codeterministic*: there exists a DWA.
- *Safety properties*: there exists a DBA where all states are final.
- *Cosafety properties*: the complement is a safety property.
- *Liveness properties*: there exists a BA where from all states there is a path to some final state lying in a strongly connected component.
- *Monitorable properties*: there exists a monitor.

According to our definition of a monitor, not both states \perp and \top need to be defined. Sometimes it is enough to see \perp or \top . For example, let $\emptyset \neq L \neq \Sigma^\omega$ be a safety property and $\mathcal{A} = (Q, \Sigma, \delta, I, Q)$ be a DBA accepting L where all states are final. Since $\emptyset \neq L$ we have $I \neq \emptyset$. Since $L \neq \Sigma^\omega$, the partially defined transition function δ is not defined everywhere. Adding a state \perp as explained above turns \mathcal{A} into a monitor \mathcal{M} for L where the state space is $Q \cup \{\perp\}$. There is no need for any state \top . The monitor \mathcal{M} also accepts L . This is however not the general case.

2 Topological properties

A topological space is a pair (X, \mathcal{O}) where X is a set and \mathcal{O} is collection of subsets of X which is closed under arbitrary unions and finite intersections. In particular, $\emptyset, X \in \mathcal{O}$. A subset $L \in \mathcal{O}$ is called *open*; and its complement $X \setminus L$ is called *closed*.

For $L \subseteq X$ we denote by \overline{L} the intersection over all closed subsets K such that $L \subseteq K \subseteq X$. It is the *closure* of L . The complement $X \setminus L$ is denoted by L^{co} .

A subset $L \subseteq X$ is called *nowhere dense* if its closure \overline{L} does not contain any open subset. The classical example of the uncountable Cantor set C inside the closed interval $[0, 1]$ is nowhere dense. It is closed and does not have any open subset. On the other hand, the subset of rationals \mathbb{Q} inside \mathbb{R} (with the usual topology) satisfies $\overline{\mathbb{Q}} = \mathbb{R}$. Hence, \mathbb{Q} is “dense everywhere” although \mathbb{Q} itself does not have any open subset.

The *boundary* of L is sometimes denoted as $\delta(L)$; it is defined by

$$\delta(L) = \overline{L} \cap \overline{L^{\text{co}}}.$$

In a metric space $B(x, 1/n)$ denotes the *ball of radius* $1/n$. It is the set of y where the distance between x and y is less than $1/n$. A set is open if and only if it is some union of balls, and the closure of L can be written as

$$\overline{L} = \bigcap_{n \geq 1} \bigcup_{x \in L} B(x, 1/n).$$

In particular, every closed set is a countable intersection of open sets. Following the traditional notation we let F be the family of closed subsets and G be the family of open subsets. Then F_σ denotes the family of countable unions of closed subsets and G_δ denotes the family of countable intersections of open subsets. We have just seen $F \subseteq G_\delta$, and we obtain $G \subseteq F_\sigma$ by duality. Since G_δ is closed under finite union, $G_\delta \cap F_\sigma$ is Boolean algebra which contains all open and all closed sets.

In this paper we deal mainly with ω -regular sets. These are subsets of Σ^ω ; and Σ^ω is endowed with a natural topology where the open sets are defined by the sets of the form $W\Sigma^\omega$ where $W \subseteq \Sigma^*$. It is called the *Cantor topology*. The Cantor topology corresponds to a complete ultra metric space: for example, we let $d(\alpha, \beta) = 1/n$ for $\alpha, \beta \in \Sigma^\omega$ where $n - 1 \in \mathbb{N}$ is the length of a maximal common prefix of α and β . (The convention is $0 = 1/\infty$.)

The following dictionary translates notation about ω -regular sets into its topological counterpart.

- Safety = closed sets = F .
- Cosafety = open sets = G .
- Liveness = *dense* = closure is Σ^ω .
- Deterministic = G_δ , see [11].
- Codeterministic = F_σ , by definition and the previous line.
- Deterministic and simultaneously codeterministic = $G_\delta \cap F_\sigma$, by definition.
- Monitorable = the boundary is nowhere dense, see [4].

Monitorability depends on the ambient space X . Imagine we embed \mathbb{R} into the plane \mathbb{R}^2 in a standard way. Then \mathbb{R} is a line which is nowhere dense in \mathbb{R}^2 . As a consequence every subset $L \subseteq \mathbb{R}$ is monitorable in \mathbb{R}^2 . The same phenomenon happens for ω -regular languages. Consider the embedding of $\{a, b\}^\omega$ into $\{a, b, c\}^\omega$ by choosing a third letter c . Then $\{a, b\}^\omega$ is nowhere dense in $\{a, b, c\}^\omega$ and hence, every subset $L \subseteq \{a, b\}^\omega$ is monitorable in $\{a, b, c\}^\omega$. The monitor has 3 states. One state is initial and by reading c we switch into the state \perp . The state \top can never be reached. In some sense this 3-state minimalistic monitor is useless: it tells us almost nothing about the language. Therefore the smallest possible monitor is rarely the best one.

Remark 1. In our setting many languages are monitorable because there exists a “forbidden factor”, for example a letter c in the alphabet which is never used. More precisely, let $L \subseteq \Sigma^\omega$ be any subset and assume that there exists a finite word $f \in \Sigma^*$ such that either $\Sigma^* f \Sigma^\omega \subseteq L$ or $\Sigma^* f \Sigma^\omega \cap L = \emptyset$. Then L is monitorable. Indeed, the monitor just tries to recognize $\Sigma^* f \Sigma^\omega$. Its size is $|f| + 2$ and can be constructed in linear time from f by algorithms of Matiyasevich [15] or Knuth-Morris-Pratt [9].

3 Constructions of monitors

Remark 1 emphasizes that one should not try simply to minimize monitors. The challenge is to construct “useful” monitors. In the extreme, think that we encode

a language L in printable ASCII code, hence it is a subset of $\{0,1\}^*$. But even in using a 7-bit encoding there were 33 non-printable characters. A monitor can choose any of them and then waits patiently whether this very special encoding error ever happens. This might be a small monitor, but it is of little interest. It does not even check all basic syntax errors.

3.1 Monitors for ω -regular languages in $G_\delta \cap F_\sigma$

The ω -regular languages in $G_\delta \cap F_\sigma$ are those which are deterministic and simultaneously codeterministic. In every complete metric space (as for example the Cantor space Σ^ω) all sets in $G_\delta \cap F_\sigma$ have a boundary which is nowhere dense. Thus, deterministic and simultaneously codeterministic languages are monitorable by a purely topological observation, see [4].

Recall that there is another characterization of ω -regular languages in $G_\delta \cap F_\sigma$ due to Staiger, [21]. It says that these are the languages which are accepted by some DWA, thus by some DBA where in every strongly connected component either all states are final or none is final.

In every finite directed graph there is at least one strongly connected component which cannot be left anymore. In the minimal DWA (which exists and which is unique and where, without restriction, the transition function is totally defined) these end-components consist of a single state which can be identified either with \perp or with \top . Thus, the DWA is itself a monitor. Here we face the problem that this DWA might be very large and also too complicated for useful monitoring.

3.2 General constructions

Let $w \in \Sigma^*$ be any word. Then the language $L = w\Sigma^\omega$ is *clopen* meaning simultaneously open and closed. The minimal monitor for $w\Sigma^\omega$ must read the whole word w before it can make a decision; and the minimal monitor has exactly $|w| + 2$ states. On the other hand, its boundary, $\bar{L} \cap \bar{L}^{\text{co}}$ is empty and therefore nowhere dense. This suggests that deciding monitorability might be much simpler than constructing a monitor. For deciding we just need any DBA accepting the safety property $\bar{L} \cap \bar{L}^{\text{co}}$. Then we can see on that particular DBA whether L is monitorable, although this particular DBA might be of no help for monitoring. Phrased differently, there is no bound between the size of a DBA certifying that L is monitorable and the size of an actual monitor for L .

Indeed, the standard construction for a monitor M_L is quite different from a direct construction of the DBA for the boundary, see for example [4]. The construction for the monitor M_L is as follows. Let $L \subseteq \Sigma^\omega$ be monitorable and given by some BA. First, we construct two DBAs: one DBA with state set Q_1 , for the closure \bar{L} and another one with state set Q_2 for the closure of the complement \bar{L}^{co} . We may assume that in both DBAs all states are final and reachable from a unique initial state q_{01} and q_{02} , respectively. Second, let $Q' = Q_1 \times Q_2$. Now, if we are in a state $(p, q) \in Q'$ and we want to read a letter $a \in \Sigma$, then exactly one out of the three possibilities can happen.

1. The states $p \cdot a$ and $q \cdot a$ are defined, in which case we let $(p, q) \cdot a = (p \cdot a, q \cdot a)$.
2. The state $p \cdot a$ is not defined, in which case we let $(p, q) \cdot a = \perp$.
3. The state $q \cdot a$ is not defined, in which case we let $(p, q) \cdot a = \top$.

Here \perp and \top are new states. Moreover, we let $q \cdot a = q$ for $q \in \{\perp, \top\}$ and $a \in \Sigma$. Hence, the transition function is totally defined. Finally, we let $Q \subseteq Q' \cup \{\perp, \top\}$ be the subset which is reachable from the initial state (q_{01}, q_{02}) . Since L is monitorable, $Q \cap \{\perp, \top\} \neq \emptyset$; and Q defines a set of a monitor \mathcal{M}_L . Henceforth, the monitor \mathcal{M}_L above is called a *standard monitor for L* . The monitor has exactly one initial state. From now on, for simplicity, we assume that every monitor \mathcal{M} has exactly one initial state and that the transition function is totally defined. Thus, we can denote a monitor \mathcal{M} as a tuple

$$\mathcal{M} = (Q, \Sigma, \delta, q_0, \perp, \top). \quad (2)$$

Here, $\delta : Q \times \Sigma \rightarrow Q$, $(p, a) \mapsto p \cdot a$ is the transition function, q_0 is the unique initial state, \perp and \top are distinguished states with $Q \cap \{\perp, \top\} \neq \emptyset$.

Definition 1. Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, \perp, \top)$, $\mathcal{M}' = (Q', \Sigma, \delta', q'_0, \perp, \top)$ be monitors. A morphism between \mathcal{M} and \mathcal{M}' is mapping $\varphi : Q \cup \{\perp, \top\} \rightarrow Q' \cup \{\perp, \top\}$ such that $\varphi(q_0) = q'_0$, $\varphi(\perp) = \perp$, $\varphi(\top) = \top$, and $\varphi(p \cdot a) = \varphi(p) \cdot a$ for all $p \in Q$ and $a \in \Sigma$.

If φ is surjective, then φ is called an epimorphism.

Another canonical monitor construction uses the classical notion of right-congruence. A *right-congruence* for the monoid Σ^* is an equivalence relation \sim such that $x \sim y$ implies $xz \sim yz$ for all $x, y, z \in \Sigma^*$. There is a canonical right-congruence \sim_L associated with every ω -language $L \subseteq \Sigma^\omega$: for $x \in \Sigma^*$ denote by $L(x) = \{\alpha \in \Sigma^\omega \mid x\alpha \in L\}$ the *quotient* of L by x . Then defining \sim_L by $x \sim_L y \iff L(x) = L(y)$ yields a right-congruence. More precisely, Σ^* acts on the set of quotients $Q_L = \{L(x) \mid x \in \Sigma^*\}$ on the right, and the formula for the action becomes $L(x) \cdot z = L(xz)$. Note that this is well-defined. This yields the *associated automaton* [21, Section 2]. It the finite deterministic transition system with state set Q_L and arcs $(L(x), a, L(xa))$ where $x \in \Sigma^*$ and $a \in \Sigma$.

There is a canonical initial state $L = L(1)$, but unlike in the case of regular sets over finite words there is no good notion of final states in Q_L for infinite words. The right congruence is far too coarse to recognize L , in general. For example, consider the deterministic language L of “infinitely many a ’s” in $\{a, b\}^\omega$. For all x we have $L = L(x)$, but in order to recognize L we need two states.

It is classical that if L is ω -regular, then the set Q_L is finite, but the converse fails badly [21, Section 2]: there are uncountably many languages where $|Q_L| = 1$. To see this define for each $\alpha \in \Sigma^\omega$ a set

$$L_\alpha = \{\beta \in \Sigma^\omega \mid \alpha \text{ and } \beta \text{ share an infinite suffix}\}.$$

All L_α are countable, but the union $\{L_\alpha \mid \alpha \in \Sigma^\omega\}$ covers the uncountable Cantor space Σ^ω . Hence, there are uncountably many L_α . However, $|Q_{L_\alpha}| = 1$ since $L_\alpha(x) = L_\alpha$ for all x .

Recall that a monitor is a DBA where the monitoring property is not defined using final states, but it is defined using the states \perp and \top . Thus, a DBA with an empty set of final states can be used as a monitor as long as \perp and \top have been assigned and the required properties for a monitor are satisfied.

Proposition 1. *Let $L \subseteq \Sigma^\omega$ be ω -regular and monitorable. Assume that L is accepted by some BA with n states. As above let $Q_L = \{L(x) \mid x \in \Sigma^*\}$ and denote $\perp = \emptyset$ and $\top = \Sigma^\omega$. Then $|Q_L| \leq 2^n$ and $Q_L \cup \{\top, \perp\}$ is the set of states for a monitor for L . At least one of the states in $\{\top, \perp\}$ is reachable from the initial state $L = L(1)$.*

The monitor in Proposition 1 with state space Q_L is denoted by \mathcal{A}_L henceforth. We say that \mathcal{A}_L is the *right-congruential* monitor for L .

Proposition 2. *Let \mathcal{A} be the right-congruential monitor for L . Then the mapping*

$$L(x) \mapsto \varphi(L(x)) = (\overline{L}(x), \overline{L^{\text{co}}}(x))$$

induces a canonical epimorphism from \mathcal{A}_L onto some standard monitor \mathcal{M}_L .

Proof. Observe that $\overline{L}(x) = \overline{L(x)}$ and $L^{\text{co}}(x) = L(x)^{\text{co}}$. Hence, $(\overline{L}(x), \overline{L^{\text{co}}}(x)) = (\overline{L(x)}, \overline{L(x)^{\text{co}}})$ and $\varphi(L(x))$ is well-defined. Now, if $\overline{L}(x) \neq \emptyset$ and $\overline{L^{\text{co}}}(x) \neq \emptyset$, then $\varphi(L(x)) \in Q$ where Q is the state space of the standard monitor \mathcal{M} . If $\overline{L}(x) = \emptyset$ then we can think that all $(\emptyset, \overline{L^{\text{co}}}(x))$ denote the state \perp ; and if $\overline{L^{\text{co}}}(x) = \emptyset$ then we can think that all $(\overline{L}(x), \emptyset)$ denote the state \top . \square

Corollary 1. *Let $L \subseteq \Sigma^\omega$ be monitorable and given by some BA with n states. Then some standard monitor \mathcal{M}_L for L has at most 2^n states.*

Proof. Without restriction we may assume that in the BA $(Q, \Sigma, \delta, I, F)$ accepting L every state $q \in Q$ leads to some final state. The usual subset construction leads first to a DBA accepting \overline{L} , where all states are final and the states of this DBA are the nonempty subsets of Q . Thus, these are $2^n - 1$ states. Adding the empty set $\emptyset = \perp$ we obtain a DBA with 2^n states where the transition function is defined everywhere. If the complement L^{co} is dense, this yields a standard monitor. In the other case we can use the subset construction also for a DBA accepting $\overline{L^{\text{co}}}$. In this case we remove all subsets $P \subseteq Q$ where $L(Q, \Sigma, \delta, P, F) = \Sigma^\omega$. (Note, for all $a \in \Sigma$ we have: if $L(Q, \Sigma, \delta, P, F) = \Sigma^\omega$ and $P' = \{q \in Q \mid \exists p \in P : (p, a, q) \in \delta\}$, then $L(Q, \Sigma, \delta, P', F) = \Sigma^\omega$, too.) Thus, if L^{co} is not dense, then the construction for a standard monitor has at most $2^n - 2$ states of the form (P, P) where $\emptyset \neq P$ and $L(Q, \Sigma, \delta, P, F) \neq \Sigma^\omega$. In addition there exists the reachable state \top and possibly the state \perp . \square

Proposition 2 leads to the question of a canonical minimal monitor, at least for a safety language where a minimal accepting DBA exists. The answer is “no” as we will see in Example 1 later.

Let us finish the section with a result on arbitrary monitorable subsets of Σ^ω which is closely related to [20, Lemma 2]. Consider any subset $L \subseteq \Sigma^\omega$ where

the set of quotients $Q_L = \{L(x) \mid x \in \Sigma^*\}$ is finite (=“zustandsendlich” or “finite state” in the terminology of [20]). If Q_L is finite, then L is monitorable if and only if the boundary is nowhere dense. In every topological space this latter condition is equivalent to the condition that the interior of L is dense in its closure \bar{L} . Translating Staiger’s result in [20] to the notion of monitorability we obtain the following fact.

Proposition 3. *Let $L \subseteq \Sigma^\omega$ be any monitorable language and let \mathcal{M} be a monitor for L with n states. Then there exists a finite word w of length at most $(n-1)^2$ such that for all $x \in \Sigma^*$ we have either $xw\Sigma^\omega \subseteq L$ or $xw\Sigma^\omega \cap L = \emptyset$.*

Proof. We may assume that $n \geq 1$ and that the state space of \mathcal{M} is included in $\{1, \dots, n-1, \perp, \top\}$. Merging \top and \perp into a single state 0 we claim that there is a word w of length at most $(n-1)^2$ such that $q \cdot w = 0$ for all $0 \leq q \leq n-1$. Since L is monitorable, there is for each $q \in \{0, \dots, n-1\}$ a finite word v_q of length at most $n-1$ such that $q \cdot v_q = 0$. By induction on k we may assume that there is a word w_k of length at most $k(n-1)$ such that for each $q \in \{0, \dots, k\}$ we have $q \cdot w_k = 0$. (Note that the assertion trivially holds for $k = 0$.) If $k \geq n-1$ we are done: $w = w_{n-1}$. Otherwise consider the state $q = k+1$ and the state $p = q \cdot w_k$. Define the word w_{k+1} by $w_{k+1} = w_k v_p$. Then the length of w_{k+1} is at most $(k+1)(n-1)$. Since w_k is a prefix of w_{k+1} and since $0 \cdot v = v$ for all v , we have $q \cdot w_{k+1} = 0$ for all $0 \leq q \leq k+1$. \square

Remark 2. The interest in Proposition 3 is that monitorability can be characterized by a single alternation of quantifiers. Instead of saying that

$$\forall x \exists w (\forall \alpha : xw\alpha \in L) \vee (\forall \alpha : xw\alpha \notin L)$$

it is enough to say

$$\exists w \forall x (\forall \alpha : xw\alpha \in L) \vee (\forall \alpha : xw\alpha \notin L).$$

The length bound $(n-1)^2$ is not surprising. It confirms Černý’s Conjecture in the case of monitors. (See [24] for a survey on Černý’s Conjecture.) Actually, in the case of monitors with more than 3 states the estimation of the length of the “reset word” is not optimal. For example in the proof of Proposition 3 we can choose the word v_1 to be a letter, because there must be a state with distance at most one to 0. The precise bound is $\binom{n+1}{2} = (n+1)n/2$ if the alphabet is allowed to grow with n [18, Theorem 6.1]. If the alphabet is fixed, then the lower bound for the length of w is still in $n^2/4 + \Omega(n)$ [14].

4 Monitorable deterministic languages

The class of monitorable languages form a Boolean algebra and every ω -regular set L can be written as a finite union $L = \bigcup_{i=1}^n L_i \setminus K_i$ where the L_i and K_i are deterministic ω -regular, [23]. Thus, if L is not monitorable, then one of the deterministic L_i or K_i is not monitorable. This motivates to study monitorable deterministic languages more closely.

Definition 2. Let $L \subseteq \Sigma^\omega$ be deterministic ω -regular. A deterministic Büchi monitor (DBM for short) for L is a tuple

$$\mathcal{B} = (Q, \Sigma, \delta, q_0, F, \perp, \top)$$

where $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ is a DBA with $L = L(\mathcal{A})$ and where $(Q, \Sigma, \delta, q_0, \perp, \top)$ is a monitor in the sense of Equation (2) for L .

The next proposition justifies the definition.

Proposition 4. Let $L \subseteq \Sigma^\omega$ be any subset. Then L is a monitorable deterministic ω -regular language if and only if there exists a DBM for L .

Proof. The direction from right to left is trivial. Thus, let L be monitorable and let $L = L(\mathcal{A})$ for some DBA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where all states are reachable from the initial state q_0 . For a state $p \in Q$ let $L(p) = L(Q, \Sigma, \delta, p, F)$. If $L(p) = \emptyset$, then $L(p \cdot a) = \emptyset$; and if $L(p) = \Sigma^\omega$, then $L(p \cdot a) = \Sigma^\omega$. Thus, we can merge all states p with $L(p) = \emptyset$ into a single non-final state \perp ; and we can merge all all states p with $L(p) = \Sigma^\omega$ into a single final state \top without changing the accepted language. All states are of the form $q_0 \cdot x$ for some $x \in \Sigma^*$; and, since L is monitorable, for each x either there is some y with $xy\Sigma^\omega \cap L = \emptyset$ or there is some y with $xy\Sigma^\omega \subseteq L$ (or both). In the former case we have $q_0 \cdot xy = \perp$ and in the latter case we have $q_0 \cdot xy = \top$. \square

Corollary 2. Let $L \subseteq \Sigma^\omega$ be a monitorable deterministic ω -regular language and \mathcal{A} be a DBA with n states accepting L . Let \mathcal{B} be a DBM for L with state set $Q_{\mathcal{B}}$ where the size of $Q_{\mathcal{B}}$ is as small as possible. Let further $Q_{\mathcal{R}}$ (resp. $Q_{\mathcal{M}}$) be the state set of the congruential (resp. smallest standard) monitor for L . Then we have

$$n \geq |Q_{\mathcal{B}}| \geq |Q_{\mathcal{R}}| \geq |Q_{\mathcal{M}}|.$$

Example 1. Let $\Sigma = \{a, b\}$ and $\Gamma = \{a, b, c, d\}$.

1. For $n \in \mathbb{N}$ consider $L = a^n b \Sigma^\omega \setminus \Sigma^* b b \Sigma^\omega$. It is a safety property. Hence, we have $\overline{L} = L$. Moreover, $\Sigma^* b b \Sigma^\omega$ is a liveness property (i.e., dense). Hence $\overline{L}^{\text{co}} = \Sigma^\omega$. It follows that the standard monitor is just the minimal DBA for L augmented by the state \perp . There are exactly $n + 4$ right-congruence classes defined by prefixes of the words $a^n b a$ and $a^n b^2$. We have $L(a^n b^2) = \emptyset$. Hence reading $a^n b^2$ leads to the state \perp . This, shows that the inequalities in Corollary 2 become equalities in that example. On the other hand b^2 is a forbidden factor for L . Hence there is a 3 state monitor for L . Still there is no epimorphism from the standard monitor onto that monitor, since in the standard monitor we have $L(a^{n+1}) = \emptyset$ but in the 3-state monitor \perp has not an incoming arc labeled by a .
2. Every monitor for the language $\Sigma^*(bab \cup b^3)\Sigma^\omega$ has at least 4 states. There are three monitors with 4 states which are pairwise non-isomorphic.

3. Let $L = (b^*a)^\omega \cup \{a, b\}^*c\{a, b, c\}^\omega \subseteq \Gamma^\omega$. Then L is monitorable and deterministic, but not codeterministic. Its minimal DBM has 4 states, but the congruential monitor $Q_{\mathcal{R}}$ has 3 states, only. We have $\bar{L} = \{a, b, c\}^\omega$ and $\bar{L}^{\text{co}} = \Gamma^\omega$. Hence, the smallest standard monitor has two states. In particular, we have $|Q_{\mathcal{B}}| > |Q_{\mathcal{R}}| > |Q_{\mathcal{M}}|$, see also Figure 1.

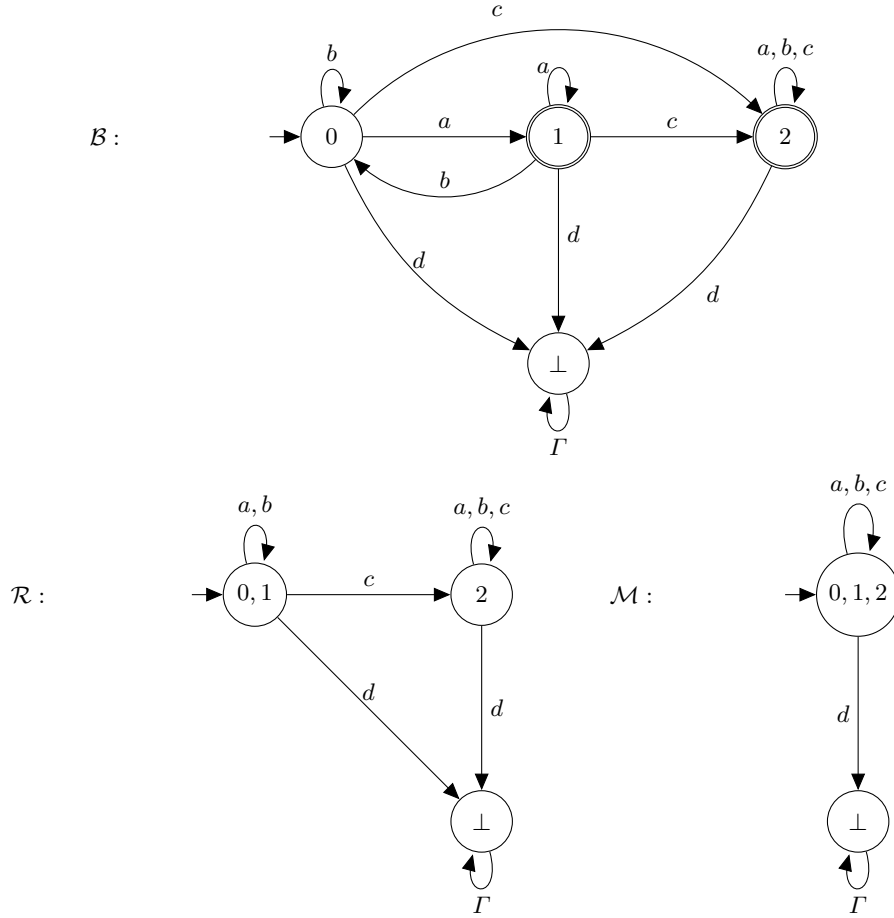


Fig. 1. Monitors \mathcal{B} , \mathcal{R} , \mathcal{M} for $L = L(\mathcal{B})$.

5 Deciding liveness and monitorability

5.1 Decidability for Büchi automata

It is well-known that decidability of liveness (monitorability resp.) is PSPACE-complete for Büchi automata. The following result for liveness is classic, for monitorability it was shown in [5].

Proposition 5. *The following two problems are PSPACE-complete:*

- **Input:** A Büchi automaton $\mathcal{A} = (Q, \Sigma, \delta, I, F)$.
- **Question 1:** Is the accepted language $L(\mathcal{A}) \subseteq \Sigma^\omega$ live?
- **Question 2:** Is the accepted language $L(\mathcal{A}) \subseteq \Sigma^\omega$ monitorable?

Proof. Both problems can be checked in PSPACE using standard techniques. We sketch this part for monitorability. The procedure considers, one after another, all subsets P such that P is reachable from I by reading some input word. For each such P the procedure guesses some P' which is reachable from P . It checks that either $L(\mathcal{A}') = \emptyset$ or $L(\mathcal{A}') = \Sigma^\omega$, where $\mathcal{A}' = (Q, \Sigma, \delta, P', F)$. If both tests fail then the procedure enters a rejecting loop.

If, on the other hand, the procedure terminates after having visited all P , then $L(\mathcal{A})$ is monitorable.

For convenience of the reader we show PSPACE-hardness of both problems by adapting the proof in [5].

We reduce the universality problem for non-deterministic finite automata (NFA) to both problems. The universality problem for NFA is well-known to be PSPACE-complete.

Start with an NFA $\mathcal{A} = (Q', \Gamma, \delta', q_0, F')$ where $\Gamma \neq \emptyset$. We use a new letter $b \notin \Gamma$ and we let $\Sigma = \Gamma \cup \{b\}$.

We will construct Büchi automata \mathcal{B}_1 and \mathcal{B}_2 as follows. We use three new states d, e, f and we let $Q = Q' \cup \{d, e, f\}$, see Figure ???. The initial state is the same as before: q_0 . Next, we define δ . We keep all arcs from δ' and we add the following new arcs.

- $q \xrightarrow{b} d \xrightarrow{a} e \xrightarrow{a} e$ for all $q \in Q' \setminus F'$ and all $a \in \Gamma$.
- $e \xrightarrow{b} d \xrightarrow{b} d$
- $q \xrightarrow{b} f \xrightarrow{c} f$ for all $q \in F'$ and all $c \in \Sigma$.

Let us define two final sets of states: $F_1 = \{f\}$ and $F_2 = \{d, f\}$. Thus, we have constructed Büchi automata \mathcal{B}_1 and \mathcal{B}_2 where

$$\mathcal{B}_i = (Q, \Gamma, \delta, q_0, F_i) \text{ for } i = 1, 2.$$

For the proof of the proposition it is enough to verify the following two claims which are actually more precise than needed.

1. The language $L(\mathcal{B}_1)$ is monitorable. It is live if and only if $L(\mathcal{A}) = \Gamma^*$.
2. The language $L(\mathcal{B}_2)$ is live. It is monitorable if and only if $L(\mathcal{A}) = \Gamma^*$.

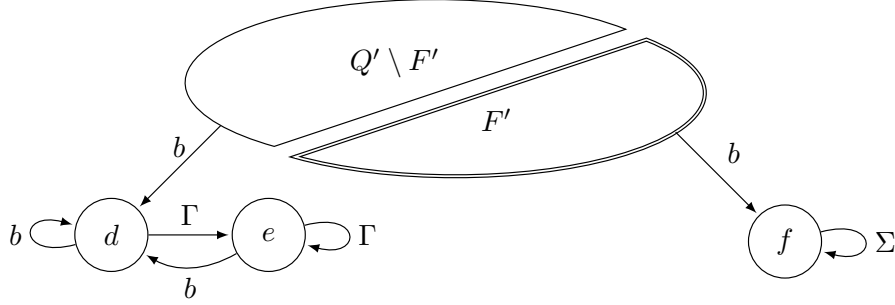


Fig. 2. PSPACE-hardness for liveness and monitorability for Büchi automata.

If $L(\mathcal{A}) = \Gamma^*$, then we have $L(\mathcal{B}_1) = L(\mathcal{B}_2) = \Sigma^\omega$, so both languages are live and monitorable.

If $L(\mathcal{A}) \neq \Gamma^*$, then there exists some word $u \notin L(\mathcal{A})$ and hence reading ub we are necessarily in state d . It follows that $ub\Sigma^\omega \cap L(\mathcal{B}_1) = \emptyset$ and $L(\mathcal{B}_1)$ is not live. Still, $L(\mathcal{B}_1)$ is monitorable. Now, for all $w \in \Sigma^*$ we have $wb^\omega \in L(\mathcal{B}_2)$. Hence, $L(\mathcal{B}_2)$ is live. However, if $u \notin L(\mathcal{A})$, then after reading ub we are in state d . Now, choose some letter $a \in \Gamma$. For all $v \in \Sigma^*$ we have $ubva^\omega \notin L(\mathcal{B}_2)$, but $ubvb^\omega \in L(\mathcal{B}_2)$. Hence, if $L(\mathcal{A}) \neq \Gamma^*$, then $L(\mathcal{B}_2)$ is not monitorable. \square

5.2 Decidability for LTL

We use the standard syntax and semantics of the linear temporal logic LTL for infinite words over some finite nonempty alphabet Σ . We restrict ourselves the pure future fragment and the syntax of $\text{LTL}_\Sigma[\text{XU}]$ is given as follows.

$$\varphi ::= \top \mid a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \text{XU } \varphi,$$

where a ranges over Σ . The binary operator XU is called the *next-until* modality.

In order to give the semantics we identify each $\varphi \in \text{LTL}_\Sigma$ with some first-order formula $\varphi(x)$ in at most one free variable. The identification is done as usual by structural induction. The formula a becomes $a(x) = P_a(x)$, where $P_a(x)$ is the unary predicate saying that the label of position x is the letter a . The formula “ φ *next-until* ψ ” is defined by:

$$(\varphi \text{XU } \psi)(x) = \exists z : (x < z \wedge \psi(z) \wedge \forall y : \varphi(y) \vee y \leq x \vee z \leq y).$$

Finally let $\alpha \in \Sigma^\omega$ be an infinite word with the first position 0, then we define $\alpha \models \varphi$ by $\alpha \models \varphi(0)$; and we define

$$L(\varphi) = \{\alpha \in \Sigma^\omega \mid \alpha \models \varphi\}.$$

Languages of type $L(\varphi)$ are called LTL *definable*. It is clear that every LTL definable language is first-order definable; and Kamp’s famous theorem [8] states

the converse. In particular, given $L(\varphi)$ there exists a BA \mathcal{A} such that $L(\varphi) = L(\mathcal{A})$. There are examples where the size of the formula φ is exponentially smaller than the size of any corresponding BA \mathcal{A} .

For a survey on first-order definable languages we refer to [3]. By LTL decidability of a property \mathcal{P} we mean that the input is a formula $\varphi \in \text{LTL}_\Sigma$ and we ask whether property \mathcal{P} holds for $L(\varphi)$. By Proposition 5 we obtain straightforwardly the following lower and upper bounds for the LTL decidability of monitorability and liveness.

Remark 3. The following two problems are PSPACE-hard and can be solved in EXPSPACE:

- **Input:** A formula $\varphi \in \text{LTL}_\Sigma$.
- **Question 1:** Is the accepted language $L(\varphi) \subseteq \Sigma^\omega$ live?
- **Question 2:** Is the accepted language $L(\varphi) \subseteq \Sigma^\omega$ monitorable?

Remark 3 is far from satisfactory since there is huge gap between PSPACE-hardness and containment in EXPSPACE. Very unfortunately, we were not able to make the gap any smaller for monitorability. There was some belief in the literature that, at least, LTL liveness can be tested in PSPACE, see for example [16]. But surprisingly this last assertion is wrong: testing LTL liveness is EXPSPACE-complete!

Proposition 6. *Deciding LTL liveness is EXPSPACE-complete:*

- **Input:** A formula $\varphi \in \text{LTL}_\Sigma$.
- **Question** *Is the accepted language $L(\varphi) \subseteq \Sigma^\omega$ live?*

EXPSPACE-completeness of liveness was proved by Muscholl and Walukiewicz in 2012, but never published. Independently, it was proved by Orna Kupferman and Gal Vardi in [10].

We give a proof of Proposition 6 in Sections 5.3 and 5.4 below. We also point out why the proof technique fails to say anything about the hardness to decide monitorability. Our proof for Proposition 6 is generic. This means that we start with a Turing machine M which accepts a language $L(M) \subseteq \Gamma^*$ in EXPSPACE. We show that we can construct in polynomial time a formula $\varphi(w) \in \text{LTL}_\Sigma$ such that

$$w \in L(M) \iff L(\varphi(w)) \subseteq \Sigma^\omega \text{ is not live.}$$

5.3 Encoding EXPSPACE computations

For the definition of Turing machines we use standard conventions, very closely to the notation e.g. in [7]. Let $L = L(M)$ be accepted by a deterministic Turing machine M , where M has set of states Q and the tape alphabet is Γ containing a “blank” symbol B . We assume that for some fixed polynomial $p(n) \geq n+2$ the machine M uses on an input word $w \in (\Gamma \setminus \{B\})^*$ of length n strictly less space than $2^N - 2$, where $N = p(n)$. (It does not really matter that M is deterministic.) Configurations are words from $\Gamma^*(Q \times \Gamma)\Gamma^*$ of length precisely 2^N , where the

head position corresponds to the symbol from $Q \times \Gamma$. For technical reasons we will assume that the first and the last symbol in each configuration is B . Let $A = \Gamma \cup (Q \times \Gamma)$.

If the input is nonempty word $w = a_1 \cdots a_n$ where the a_i are letters, then the initial configuration is defined here as

$$C_0 = B(q_0, a_1)a_2 \cdots a_n \underbrace{BBBBB \cdots B}_{2^N - n - 1 \text{ times}}.$$

For $t \geq 0$ let C_t be configuration of M at time t during the computation starting with the initial configuration C_0 on input w . We may assume that the computation is successful if and only if there is some t such that a special symbol, say q_f , appears in C_t . Thus, we can write each C_t as a word $C_t = a_{0,t} \cdots a_{m,t}$ with $m = 2^N - 1$; and we have $w \in L(M)$ if and only if there are some $i \geq 1$ and $t \geq 1$ such that $a_{i,t} = q_f$.

In order to check that a sequence C_0, C_1, \dots is a valid computation we may assume that the Turing machine comes with a table $\Delta \subseteq A^4$ such that the following formula holds:

$$\forall t > 0 \forall 1 \leq i < 2^N - 1 : (a_{i-1,t-1}, a_{i,t-1}, a_{i+1,t-1}, a_{i,t}) \in \Delta.$$

Without restriction we have $(B, B, B, B) \in \Delta$, because otherwise M would accept only finitely many words.

We may express that we can reach a final configuration C_t by saying:

$$\exists t \geq 1 \exists 1 \leq i < 2^N : a_{i,t} = q_f.$$

As in many EXPSPACE-hardness proofs, for comparing successive configurations we need to switch to a slightly different encoding, by adding the tape position after each symbol from A . To do so, we enlarge the alphabet A by new symbols $0, 1, \$, \#, k_1, \dots, k_N$ which are not used in any C_t so far. Hence, $\Sigma = A \cup \{0, 1, \$, \#, k_1, \dots, k_N\}$. We encode a position $0 \leq i < 2^N$ by using its binary representation with exactly N bits. Thus, each i is written as a word $\text{bin}(i) = b_1 \cdots b_N$ where each $b_p \in \{0, 1\}$. In particular, $\text{bin}(0) = 0^N$, $\text{bin}(1) = 0^{N-1}1, \dots, \text{bin}(2^N - 1) = 1^N$.

Henceforth, a configuration $C_t = a_{0,t} \cdots a_{m,t}$ with $m = 2^N - 1$ is encoded as a word

$$c_t = a_{0,t} \text{bin}(0) \cdots a_{m,t} \text{bin}(m) \$.$$

Words of this form are called *stamps* in the following. Each stamp has length $2^N \cdot N + 1$. If a factor $\text{bin}(i)$ occurs, then either $i = m$ (i.e., $\text{bin}(i) = 1^N$) and the next letter is $\$$ or $i < m$ and the next letter is some letter from the original alphabet A followed by the word $\text{bin}(i + 1)$.

Now we are ready to define a language $L = L(w)$ which has the property that L is not live if and only if $w \in L(M)$. We describe the words $\alpha \in \Sigma^\omega$ which belong to L as follows.

1. Assume that α does not start with a prefix of the form $c_0 \cdots c_\ell \#$, where c_0 corresponds to the initial configuration w.r.t. w , each c_t is a stamp and in the stamp c_ℓ the symbol q_f occurs. Then α belongs to L .

2. Assume now that α starts with a prefix $c_0 \cdots c_\ell \#$ as above. Then we let α belong to L if and only if the set of letters occurring infinitely often in α witness that the prefix $c_0 \cdots c_\ell$ of stamps is **not** a valid computation. Thus, we must point to some $t \geq 1$ and some position $1 \leq i < m$ such that $(a_{i-1,t-1}, a_{i,t-1}, a_{i+1,t-1}, a_{i,t}) \notin \Delta$. The position i is given as $\text{bin}(i) = b_1 \cdots b_N \in \{0, 1\}^N$. The string $\text{bin}(i)$ defines a subset of Σ :

$$I(i) = \{k_p \in \{k_1, \dots, k_N\} \mid b_p = 1\}.$$

The condition for α to be in L is that for some t the mistake from c_{t-1} to c_t is reported by $(a_{i-1,t-1}, a_{i,t-1}, a_{i+1,t-1}, a_{i,t}) \notin \Delta$ and the position i such that $I(i)$ equals the set of letters k_p which appear infinitely often in α . Note that since we excluded mistakes at position $i = 0$ (because of the leftmost B), the set $I(i)$ is non-empty.

Lemma 1. *The language $L = L(w)$ is not live if and only if $w \in L(M)$.*

Proof. First, let $w \in L(M)$. Then we claim that L is not live. To see this let $u = c_0 \cdots c_\ell \#$, where the prefix $c_0 \cdots c_\ell$ is a valid accepting computation of M . There is no mistake in $c_0 \cdots c_\ell$. Thus we have $u\Sigma^\omega \cap L = \emptyset$, so indeed, L is not live.

Second, let $w \notin L(M)$. We claim that L is live. Consider any $u \in \Sigma^*$. Assume first that u does not start with a prefix of the form $c_0 \cdots c_\ell \#$, where c_0 corresponds to the initial configuration w.r.t. w , each c_t is a stamp and in the stamp c_ℓ the symbol q_f occurs. Then we have $u\Sigma^\omega \subseteq L$.

Otherwise, assume that $c_0 \cdots c_\ell \#$ is a prefix of u and that all c_t 's are stamps, with c_0 initial and c_ℓ containing q_f . There must be some mistake in $c_0 \cdots c_\ell \#$, say for some i and t . Let $I(i)$ be as defined above. As $i \geq 1$ we have $I(i) \neq \emptyset$. Therefore we let β be any infinite word where the set of letters appearing infinitely often is exactly the set $I(i)$. By definition of L we have $u\beta \in L$. Hence, L is live. \square

There are other ways to encode EXPSPACE computations which may serve to prove Proposition 6, see for example [10]. However, these proofs do not reveal any hardness for LTL monitorability. In particular, they do not reveal EXPSPACE or EXPTIME hardness. For our encoding this can be made very precise.

Remark 4. Since we are interested in EXPSPACE-hardness, we may assume that there are infinitely many w with $w \notin L(M)$. Let n be large enough, say $n \geq 3$ and $w \notin L(M)$, then $(B, (q_0, a_1), a_2, q_f) \notin \Delta$, where $w = a_1 a_2 \cdots$ because otherwise $w \in L(M)$. Define c_1 just as the initial stamp c_0 with the only difference that the letter (q_0, a_1) is replaced by the symbol q_f . Let $u = c_0 c_1 \#$, then for every $v \in \Sigma^*$ we have that $uv(k_N)^\omega \in L$ (i.e., there is a mistake at position 1), but $uv(k_1 k_2 \cdots k_N)^\omega \cap L = \emptyset$ (i.e., there is no mistake at position $2^N - 1$) because $(B, B, B, B) \in \Delta$. Thus, L is not monitorable.

5.4 Proof of Proposition 6

LTL liveness is in EXPSPACE by Remark 3. The main ideas for the proof are in the previous subsection. We show that we can construct in polynomial time on input w some $\varphi \in \text{LTL}_\Sigma$ such that $L(\varphi) = L(w)$. This can be viewed as a standard exercise in LTL. The solution is a little bit tedious and leads to a formula of at most quadratic size in n . The final step in the proof is to apply Lemma 1. \square

6 Conclusion and outlook

In the paper we studied monitorable languages from the perspective of what is a “good monitor”. In some sense we showed that there is no final answer yet, but monitorability is a field where various interesting questions remain to be answered.

Given an LTL formula for a monitorable property one can construct monitors of at most doubly exponential size; and there is some indication that this is the best we can hope for, see [2]. Still, we were not able to prove any hardness for LTL monitorability beyond PSPACE. This does not mean anything, but at least in theory, it could be that LTL monitorability cannot be tested in EXPTIME, but nevertheless it is not EXPTIME-hard.

There is also another possibility. Deciding monitorability might be easier than constructing a monitor. Remember that deciding monitorability means to test that the boundary is nowhere dense. However we have argued that a DBA for the boundary does not give necessarily any information about a possible monitor, see the discussion at the beginning of Section 3.2.

A more fundamental question is about the notion of monitorability. The definition is not robust in the sense that every language becomes monitorable simply by embedding the language into a larger alphabet. This is somewhat puzzling, so the question is whether a more robust and still useful notion of monitorability exist.

Finally, there is an interesting connection to learning. In spite of recent progress to learn general ω -regular languages by [1] it is not known how to learn a DBA for deterministic ω -regular languages in polynomial time. The best result is still due to Maler and Pnueli in [13]. They show that it is possible to learn a DWA for a ω -regular language L in $G_\delta \cap F_\sigma$ in polynomial time. The queries to the oracle are membership question “ $uv^\omega \in L?$ ” where u and v are finite words and the query whether a proposed DWA is correct. If not, the oracle provides a shortest counterexample of the form uv^ω .

Since a DWA serves also as a monitor we can learn a monitor the very same way, but beyond $G_\delta \cap F_\sigma$ it is not known that membership queries to L and queries whether a proposed monitor is correct suffice. As a first step one might try find out how to learn a deterministic Büchi monitor in case it exists. This is a natural class beyond $G_\delta \cap F_\sigma$ because canonical minimal DBA for these languages exist. Moreover, just as for DWA this minimal DBA is an DBM, too.

Another interesting branch of research is monitorability in a distributed setting. A step in this direction for infinite Mazurkiewicz traces was outlined in [5].

Acknowledgment

The work was done while the first author was visiting LaBRI at the Université Bordeaux in June 2015. The hospitality of LaBRI and their members is greatly acknowledged.

The authors thank Andreas Bauer who communicated to us (in June 2012) that the complexity of LTL-liveness should be regarded as open because published proofs stating PSPACE-completeness were not convincing. We also thank Ludwig Staiger, Gal Vardi, and Mikhail Volkov for helpful comments.

References

1. D. Angluin and D. Fisman. Learning regular omega languages. In P. Auer, A. Clark, T. Zeugmann, and S. Zilles, editors, *Algorithmic Learning Theory - 25th International Conference, ALT 2014, Bled, Slovenia, October 8-10, 2014. Proceedings*, volume 8776 of *Lecture Notes in Computer Science*, pages 125–139. Springer, 2014.
2. A. Bauer, M. Leucker, and C. Schallhart. Monitoring of real-time properties. In *Proceedings of FSTTCS’06*, number 433 in LNCS, pages 260–272. Springer, 2006.
3. V. Diekert and P. Gastin. First-order definable languages. In J. Flum, E. Grädel, and Th. Wilke, editors, *Logic and Automata: History and Perspectives*, Texts in Logic and Games, pages 261–306. Amsterdam University Press, 2008.
4. V. Diekert and M. Leucker. Topology, monitorable properties and runtime verification. *Theoretical Computer Science*, 537:29–41, 2014. Special Issue of ICTAC 2012.
5. V. Diekert and A. Muscholl. On distributed monitoring of asynchronous systems. In C.-H. L. Ong and R. J. G. B. de Queiroz, editors, *WoLLIC*, volume 7456 of *Lecture Notes in Computer Science*, pages 70–84. Springer, 2012.
6. K. Gondi, Y. Patel, and A. P. Sistla. Monitoring the full range of ω -regular properties of stochastic systems. In N. D. Jones and M. Müller-Olm, editors, *VMCAI*, volume 5403 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2009.
7. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
8. H. Kamp. *Tense Logic and the Theory of Linear Order*. PhD thesis, University of California, 1968.
9. D. Knuth, J. H. Morris, and V. Pratt. Fast pattern matching in strings. *SIAM J. Comput.*, 6:323–350, 1977.
10. O. Kupferman and G. Vardi. On relative and probabilistic finite counterabilty. In *Proceedings 24th EACSL Annual Conference on Computer Science Logic (CSL 2105)*. Springer, 2015. To appear.
11. L. H. Landweber. Decision problems for ω -automata. *Mathematical Systems Theory*, 3(4):376–384, 1969.
12. M. Leucker and C. Schallhart. A brief account of runtime verification. *Journal of Logic and Algebraic Programming*, 78(5):293–303, may/june 2009.

13. O. Maler and A. Pnueli. On the learnability of infinitary regular sets. *Inf. Comput.*, 118:316–326, 1995.
14. P. V. Martugin. A series of slowly synchronizing automata with a zero state over a small alphabet. *Information and Computation*, 206:1197–1203, 2008.
15. Yu. Matiyasevich. Real-time recognition of the inclusion relation. *Journal of Soviet Mathematics*, 1:64–70, 1973. Translated from *Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Matematicheskogo Instituta im. V. A. Steklova Akademii Nauk SSSR*, Vol. 20, pp. 104–114, 1971.
16. U. Nitsche and P. Wolper. Relative liveness and behavior abstraction (Extended abstract). In J. E. Burns and H. Attiya, editors, *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing (PODS 1997)*, Santa Barbara, California, USA, August 21–24, 1997, pages 45–52. ACM, 1997.
17. A. Pnueli and A. Zaks. PSL model checking and run-time verification via testers. In *Formal Methods*, volume 4085 of *LNCIS*, pages 573–586. Springer, 2006.
18. I. Rystsov. Reset words for commutative and solvable automata. *Theoretical Computer Science*, 172:273–279, 1997.
19. A. P. Sistla, M. Zefran, and Y. Feng. Monitorability of stochastic dynamical systems. In G. Gopalakrishnan and S. Qadeer, editors, *CAV*, volume 6806 of *Lecture Notes in Computer Science*, pages 720–736. Springer, 2011.
20. L. Staiger. Reguläre Nullmengen. *Elektronische Informationsverarbeitung und Kybernetik*, 12(6):307–311, 1976.
21. L. Staiger. Finite-state ω -languages. *Journal of Computer and System Sciences*, 27:434–448, 1983.
22. L. Staiger and K. W. Wagner. Automatentheoretische und automatenfreie Charakterisierungen topologischer Klassen regulärer Folgenmengen. *Elektronische Informationsverarbeitung und Kybernetik*, 10:379–392, 1974.
23. W. Thomas. Automata on infinite objects. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, chapter 4, pages 133–191. Elsevier Science Publishers B. V., 1990.
24. M. V. Volkov. Synchronizing automata and the Černý conjecture. In C. Martín-Vide, F. Otto, and H. Fernau, editors, *Language and Automata Theory and Applications, Second International Conference, LATA 2008, Tarragona, Spain, March 13–19, 2008. Revised Papers*, volume 5196 of *Lecture Notes in Computer Science*, pages 11–27. Springer, 2008.
25. K. W. Wagner. On omega-regular sets. *Information and Control*, 43:123–177, 1979.